View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Librarians' Digital Library

DRTC-HP International Workshop on
Building Digital Libraries using DSpace
7th - 11th March 2005
DRTC, Bangalore

Paper : M

Localization, Internationalization and DSpace

Jim Downing
ojd20@cam.ac.uk

Cambridge University, USA

Abstract

In common with most software, DSpace was written with an English speaking audience in mind. It has subsequently been adopted by users for whom English is not a first language, and to store resources in non-English languages and scripts. This broadening of language base looks set to continue.

Localization is an inevitable consequence of this broadening user base, as new users adapt the software to support content in new languages, and customize the interface so that it can be used by non-English speakers.

Without internationalization, localization efforts are isolated, difficult and often have no standard approach. In order to expedite the global distribution of DSpace, it is important to look at ways of unifying and facilitating localization efforts through internationalization.

This article will explain some approaches to internationalization and localization, and highlight some of the difficulties and complexities involved. It will describe the current capabilities of DSpace software with regard to internationalization and localization, and look ahead at how, and when, these capabilities might be enhanced.

1. Definitions

In software engineering *localization* (L10N[1]) is the process of tailoring software to local conventions and preferences, and *internationalization* is the process of writing or modifying the software such that it can be localized more easily. More formally: -

"*Localization* is the process of adapting software for a specific region or language by adding locale-specific components and translating text."

"*Internationalization* is the process of designing an application so that it can be adapted to various languages and regions without engineering changes"[1]

Digital repository software presents an added complication in that the content and associated metadata handled by a repository may in multiple languages and character sets — *multi-localization*.

2. Unicode

"Text encoding" is the process of turning a sequence of text characters or glyphs into a sequence of bytes of information, so that it can be stored or transmitted by a computer. Historically, one of the biggest difficulties in localizing software to a new language was caused by incompatible text encoding standards. Unicode aims to provide a consistent approach to encoding across all languages and character sets.

Unicode[2] (specifically UTF-8) is currently the best universal approach for encoding most text, including Indic languages. "Creation of Digital Libraries in Indian Languages Using Unicode"[3] gives a good explanation of the alternatives to Unicode that have been attempted in the Indian context.

Unicode defines encoding only in terms of characters, and leaves the rendering of the associated glyphs to language specific rules and font support. This important feature allows Unicode to efficiently handle so many languages and alphabets in a single encoding standard.

Glyphs may be constructed from multiple characters, according to language specific rules that are also part of the Unicode specification. For example, this means that Indic consonant characters carrying non-default vowels are represented as two Unicode separate character codes, which are subsequently combined when constructing the glyph. This separation of character and glyph also allows a single character code to be used where a character may have alternative glyph representations (e.g. in Arabic, where the position of the character in the word determines the glyph. See "Complex Script Rendering"[4] for a summary of the issues that occur in various different languages).

What Unicode means in terms of digital repositories is that the same encoding can be used for all text in multiple languages across a repository. As long as the relevant font support is installed on the user's computer, unicode encoded text can be directly displayed, even if a single page or document contains text in multiple languages.

1 'Localization' is often abbreviated to avoid typing. L10N is used because there are 10 characters between the initial 'L' and the terminal 'N'. Similarly 'Internationalization' is abbreviated to I18N.

2.1 Unicode and DSpace

DSpace relies on PostgreSQL to provide the encoding for content metadata. When installing DSpace, administrators must create the underlying database with UTF-8 encoding. Consequently any UTF-8 metadata added to DSpace will be preserved and can be rendered to the user *as long as the language value on the metadata field is set correctly.*

DSpace automatically preserves the original bytes of any file imported to it, so UTF-8 content submitted to DSpace will be preserved.

2.2 Issues when creating Unicode content

Creating Unicode content programmatically is straightforward in most programming languages. Creating it manually through a text editor or form inputs in a web browser is not currently so well supported. Initial research indicates that Windows XP (although none of the older versions) may provide the easiest route to generating Indic Unicode content, although setting up a PC so that it is able to generate Unicode Indic text will require some technical assistance.

Microsoft Windows XP has support for Unicode, and fonts for 9 Indian scripts. They also run Microsoft Bhasha[5], an online resource with help for developers and users in enabling Indic language computing. Script-specific keyboards are available, and keyboard overlays can be enabled in the operating system.

Linux has support for Unicode at the operating system level, with keyboard mappings similar to those in Windows. The IndLinux project[6] aims to create a distribution of Linux localized to Indic languages. Most distributions of linux include OpenType fonts for Devanagari, Gujarati, Punjabi, Kannada, Oriya and Telugu. The author's experience with these fonts is that they are occasionally incomplete (some glyphs missing). Linux also has a rendering engine (Pango[7]) for international text, but other Linux applications do not always use this by default (or at all). Some Linux text editors have better unicode support than others, Yudit[8] for example, being better than Emacs[9][2] .

Most operating system localization efforts concentrate on localizing the *whole* operating system and user interface into a new language and script. Adapting the operating system, or a particular application, to create Unicode in a language other than native localized one is often more difficult.

3. User Interface

The User Interface presented by the repository software is perhaps the most immediately obvious product of L10N. If software has not been internationalized, then localizing the software is often difficult. Text elements are usually held in an interface layer, but some messages, email templates etc. are often embedded deeper in the software and can be hard to locate and change. If the original software is still under development, it can be onerous to maintain each localization.

A better approach is to internationalize the software first. At a basic level this means arranging that all text and messages be looked up from an external source rather than embedded directly in the software. The software can then be re-localized simply by switching the external source, either statically through configuration or dynamically.

2 To the author's great frustration, the Devanagari script in this paper had to be imported as images because of difficulties in getting his favourite text editor (emacs) to generate Devanagari unicode

DSpace uses Java technology, which has some support ready built to help with application internationalization. The next sections (3.1 and 3.1.1) describe an overview of the technical details of Java I18N, and will be of particular interest to those of a technical inclination.

3.1 I18N, L10N in Java™

A step-by-step introduction to the techniques summarized here can be found in the Java Internationalization Tutorial[1], available free of charge on the web.

Internationalization in Java revolves around the `Locale` object[3] ; an object that represents the language and region of the user. `Locale` objects are created using an ISO-639[11] language code and an ISO-3166[12] region code. For example: -

```
myLocale = new Locale(en, GB);
hindiLocale = new Locale(hi, IN);
```

There are some shortcut `Locales` provided, e.g. `Locale.UK`, `Locale.FRENCH`. Unfortunately there are currently no shortcuts for India or Indian languages.

3.1.1 Resource Bundles

The locale object can be used to interact with `ResourceBundle` objects, which are the main mechanism for enabling different text and different behaviours based on language and region: -

```
ResourceBundle resources = ResourceBundle.getBundle(messages,
myLocale);
message = resources.getString(`i-can-eat-glass');
```

The value of `message` will depend on the language and region settings used to create the locale. For example, an English locale `message` might contain 'I can eat glass and it doesn't hurt me'[4] whilst with an Hindi locale it might contain 'काँच खा सकता हूँ, मुझे उस से कोई पीडा नही होती.'[5]

There are several ways of implementing resource bundles. Perhaps the simplest is to use properties files[6] . Each resource bundle can be backed with one or more properties files, the name of the files determining which one is used. For example, if you are looking for a bundle called labels and the `Locale` used has language `cy` (Welsh) and region `GB` (Great Britain), Java will look for a file called `labels_cy_GB.properties`.

In the example above you could have two files: -

```
# messages_en_GB.properties
```

3 The `Locale` and `ResourceBundle` classes can be found in the java.util package of the Java platform in versions since 1.1[10]

4 This unusual phrase was originally proposed as a more interesting first phrase to learn in a language than "Where is the bathroom, please? ". Because it has been widely translated, it has subsequently been used to sample UTF-8 support[13]

5 This phrase is as rendered by the default installation of Mozilla Firefox on Debian Linux with Indic Open Type fonts.

6 A properties file is a simple text file where each line consists of key-value pairs

```
   i-can-eat-glass = I can eat glass and it doesn't
hurt me

# messages_hi_IN.properties
   i-can-eat-glass =
   काँच खा सकता हूँ, मुझे उस से कोई पीडा नहीं होती.
```

Both files can be distributed with the software. All messages obtained via the `ResourceBundle` mechanism can be switched from one language to the other simply by using a different `Locale` when selecting them. The `Locale` used can be configured in the software at installation, or constructed dynamically for each user.

3.1.2 Other I18N support in Java™

Java also provides some support for performing the following tasks in generic way that can specialised to language and region: -

- Sorting and collating text
- Detecting word boundaries
- Determining character properties (whether a character is a digit, a letter, whitespace etc.)
- Formatting numbers and currencies (e.g. whether to use comma or point before decimal places)
- Formatting Dates and Times

3.2 I18N, L10N in DSpace

At the time of writing, DSpace 1 (the current version) uses none of the I18N features provided by Java. There are plans to introduce `Locale` based collation. The idea of internationalizing the JSP interfaces has been proposed, although this may be difficult due to the level of functionality embedded in the JSPs.

The development community have yet to make a decision on whether to make full use of these facilities in DSpace 2 (in the design stage). A key factor in this decision will be the participation of users who require I18N features in the design discussions. If you feel strongly about this, join the discussion on the `dspace-devel` mailing list[7] .

The DSpace 1 interface has been translated without I18N into French (at least three different translations exist), Portuguese, Spanish and Dutch. Translating DSpace involves creating a entire custom interface. When building DSpace from source, one can override any or all of the JSP pages that make up the interface. This comes at a price, as the JSP pages also contain some logic that may change in the default interface, and these changes will need to be periodically transferred to your custom interface.

It is also possible to customize the emails sent by DSpace by editing the templates in the `config/emails` directory. Caution should be exercised when doing this — many mail clients do not support Unicode content properly.

7 Subscribe at `http://sourceforge.net/mail/? group_id=19984`

4. Sorting and collation

4.1 Collation within a single language

Collation in Unicode is commonly misunderstood. It is often stated that Unicode does not provide proper collation because the ordering of the character codes is incorrect for a particular language, or insufficient to represent complexities in collating in a language. The observation about character code ordering is true, Unicode does handle collation correctly because it is not just a table of character codes, but also provides a customisable collation algorithm[15] that works for each language.

A limitation to Unicode's collation facilities is that it does not define how to collate mixed language sources, because "Generally, the customers using a particular collation will want text sorted uniformly, no matter what the source"[14]. This causes some unusual results.

If you have a list of titles where some titles are written in Hindi in Devanagari script and some in English, the collation will group all the Hindi titles separately from the English. The ordering of the Hindi titles will depend upon whether you used English collation rules or the Hindi language collation rules. The English collation rules will sort the English titles properly, but will use character codes to sort the Hindi titles. The safe choice in this case is to use the Hindi collator, as character code ordering will do a reasonable job of sorting English (aside from uppercase / lowercase equivalence).

DSpace 1 was not designed to support multiple languages. At the time of writing it is not possible to alter the sorting language from US English (although this feature is planned for version 1.3). Multiple languages will appear in the same browse list, grouped by character set (e.g. languages that use Latin script will be grouped together), but the ordering of titles and names in each language group will be according to the Unicode character code values.

5. Text indexing and searching

At its most basic, text indexing involves breaking a document into tokens (often words, phrases or word fragments) and storing information on the relationship between the tokens and the document (frequency, position etc). Text searching involves breaking a search query into similar tokens, calculating similar information on the relationship between the query tokens and the overall query and then using this information to calculate which documents are most 'similar' to the query. *Readings in Information Retrieval*[16] gives a comprehensive introduction to the relevant techniques.

A very basic approach will simply break each document into words using whitespace and punctuation, and record which words occur in which documents. The query would be broken into words, enabling the search to indicate which documents have words in common with the query. This approach will work for any language that uses whitespace to separate words.

There are many different, more complex approaches used to improve search accuracy and recall: -

- Using *stop lists* removes words that are not useful for search results (e.g. 'a', 'he', 'the' ...).
- *Stemming algorithms* discard multiple suffixes from associated words (e.g. 'comput' from 'computer', 'computing' and 'computation').
- *Synonym engines* inject other tokens with similar meanings.
- *Semantic engines* attempt to resolve the semantic context of words (especially homonyms) by examining the surrounding sentence and paragraph to the word.

These methods of improving text searching are all language specific. In order to improve the search in a localized repository, some language specific indexing will be needed. Moreover, a *multi*-localized repository will need to arrange to have a search index for each language of content to be searched, so that the relevant indexing algorithm variants can be used.

5.1 Text indexing in DSpace — Lucene

DSpace uses the excellent Jakarta Lucene[17] package for text indexing and searching. Lucene has highly flexible and extensible text analysis. It contains a whitespace tokenizer that can support very basic indexing in most languages, and comes with more advanced analyzers for English, German, Russian, Chinese, Brazilian, Czechoslovakian, French and Dutch[8] .

The ISI's Documentation Research & Training Centre[18] are currently working on stemming algorithms for two Indian languages, and hope to incorporate their work into Lucene in the future.

6. Conclusions

- The UTF-8 encoding of Unicode as a universal text encoding standard is the lynchpin in the efficient creation of internationalized software and digital content repositories.
- UTF-8 is well supported in the software layer, and it is possible to create applications that can handle content and metadata in a number of languages, and to adapt a user interface to new languages and regions without significant programming effort.
- UTF-8 is currently less well supported in operating systems, browsers and editing tools.
- DSpace uses Java™technology, which provides mechanisms for I18N and L10N. DSpace 1 currently does not take advantage of these mechanisms, and supports localization only through customization. DSpace will be re-engineered for version 2, and DSpace 2 is currently in the design stage. This represents an opportunity to include internationalization as a priority in the design, an opportunity that will be assisted by advocacy for I18N in the design discussion.
- The requirement to make a repository software *multi*-lingual is beyond the scope of traditional localization. There are technical and logistical issues in user interface, deployment, collation and text indexing. Some of these can be addressed, but some (such as collation issues) cannot be worked around.

7. References

1. The Sun Internationalization Tutorial.
http://java.sun.com/docs/books/tutorial/i18n/intro/index.html

2. The Unicode Standard.
http://www.unicode.org/.
The Indic scripts and languages FAQ may be of particular interest: -
http://www.unicode.org/faq/indic.html

3. ARD Prasad, *Creation of Digital Libraries in Indian Languages Using Unicode*. *https://drtc.isibang.ac.in/retrieve/72/I_unicode.pdf*

8Some languages are supported better than others, and some support requires the download of additional packages

4. Richard Ishida, *Complex Script Rendering*.
http://people.w3.org/rishida/scripts/tutorial/script-intro-2.pdf

5. Microsoft Bhasha.
http://www.bashaindia.com/

6. The IndLinux Project.
http://indlinux.org/

7. Pango. An open-source framework for the layout and rendering of internationalized text.
http://www.pango.org/

8. Yudit. Open-source Unicode text editor.
http://www.yudit.org/

9. GNU Emacs. Open-source multi-purpose text editor.
http://www.gnu.org/software/emacs/emacs.html

10. Sun Microsystems, Java 2 Platform Standard Edition 5.0 API Specification.
http://java.sun.com/j2se/1.5.0/docs/api/index.html

11. ISO 639. Code for the representation of names of languages.
http://www.oasis-open.org/cover/iso639a.html

12. ISO 3166. Standard list of country names and codes.
http://www.iso.org/iso/en/prods-services/iso3166ma/index.html

13. Frank da Cruz, UTF-8 Sampler.
http://www.columbia.edu/kermit/utf8.html

14. Unicode Collation FAQ.
http://www.unicode.org/faq/collation.html

15. Mark Davis, Ken Whistler, Unicode Collation Algorithm.
http://www.unicode.org/reports/tr10/

16. Karen Sparck Jones (Editor), Peter Willett (Editor), Peter Willet (Editor) *Readings in Information Retrieval*. ISBN:1558604545.

17. The Apache Jakarta Project, Lucene, a high performance, full-featured text search engine library.
http://jakarta.apache.org/lucene/docs/index.html

18. Indian Statistical Institute, Documentation Research & Training Centre.
http://drtc.isibang.ac.in/DRTC/